63 Java 8 Interview Questions to Ask Developers

Questions

1. Can you explain the concept of functional interfaces in Java 8?

2. How does the Stream API in Java 8 differ from collections?

3. What are the benefits of using the Optional class in Java 8?

4. How do default methods in interfaces work, and why were they introduced in Java 8?

5. Can you explain the concept of method references in Java 8?

6. What is the significance of the @FunctionalInterface annotation?

7. How does the forEach method in Java 8 differ from a traditional for loop?

8. What are some of the new Date and Time API improvements in Java 8?

9. Can you explain the difference between a stream and a parallel stream in Java 8?

10. What is a lambda expression, and how does it enhance code readability in Java 8?

11. How does the Collectors class work in Java 8, and what are some of its common methods?

12. Can you describe the use of the filter method in Java 8 streams?

13. How do you handle exceptions in lambda expressions?

14. What is the purpose of the IntStream, LongStream, and DoubleStream classes?

15. Can you explain how the map method works in Java 8 streams?

16. What are the main differences between findFirst and findAny in Java 8 streams?

17. How do you convert a stream back to a list or an array in Java 8?

18. Explain the concept of stream pipelining in Java 8.

19. What is the purpose of the flatMap method in Java 8 streams?

20. Can you discuss the difference between intermediate and terminal operations in Java 8 streams?

21. How does the reduce method work in Java 8 streams, and when would you use it?

22. What are the benefits of using the @Repeatable annotation in Java 8?

23. How do you use the Predicate functional interface in Java 8?

24. Can you describe the use of the BiFunction interface in Java 8?

25. How has the Java 8 Date and Time API improved over the older java.util.Date?

26. What are the primary uses of the java.util.concurrent package in Java 8?

27. Explain how you would utilize the Spliterator in Java 8.

28. How does the CompletableFuture class in Java 8 help with asynchronous programming?

29. Can you explain the significance of the java.util.function package in Java 8?

30. What are the main advantages of using the new Java 8 features in terms of code readability and maintainability?

31. How do you approach debugging streams in Java 8?

32. What is a Spliterator and how does it differ from an Iterator in Java 8?

33. Can you discuss how the type inference in Java 8 has evolved compared to previous versions?

34. What role do functional interfaces play in enhancing the functional programming capabilities of Java 8?

35. Explain how the new default methods in interfaces can help with API design and backward compatibility.

36. What is the purpose of the java.util.Optional class, and how does it help in handling null values?

37. How does the Collectors class enhance data collection and aggregation in Java 8?

38. Can you explain the use of the Predicate functional interface and provide a practical example?

39. How does the concept of immutability in Java 8 benefit application development, particularly in concurrent programming?

40. Can you describe the differences between the Java 8 Streams API and traditional iteration methods, and why one may be preferred over the other?

41. What are some strategies for optimizing performance when working with large data sets in Java 8 streams?

42. How do you implement custom collectors in Java 8, and when would you use them?

43. Can you explain the significance of the CompletableFuture class in handling multiple asynchronous tasks and its impact on performance?

44. What is the role of the @FunctionalInterface annotation in Java 8, and how does it facilitate functional programming?

45. How do you ensure thread safety when using mutable data structures in Java 8?

46. Can you explain how the concept of 'Optional' can be integrated into a legacy codebase effectively?

47. What are the advantages of using method references over lambda expressions in Java 8?

48. How would you use the Stream API to perform a grouping operation on a collection of objects?

49. What is the purpose of the @FunctionalInterface annotation in Java 8?

50. How does the Stream API in Java 8 help with processing collections of data?

51. What is the purpose of the Optional class in Java 8?

52. How do default methods in interfaces work in Java 8?

53. What is the significance of method references in Java 8?

54. Can you explain the concept of stream pipelining in Java 8?

55. How would you explain the benefits of using streams for performance in Java 8?

56. What are some ways to optimize the performance of lambda expressions in Java 8?

57. How can you ensure thread safety when using streams in a multi-threaded environment?

58. How does the use of the `reducè` method in Java 8 streams enhance performance?

59. What are some common pitfalls to avoid when working with Java 8 streams for performance optimization?

60. How can the `Collectorṣ` class be used to enhance performance in Java 8 streams?